



下载APP



12 | IoT Hub: 面对海量设备如何打造高性能设备接入层?

2020-12-04 郭朝斌

物联网开发实战

进入课程 >



讲述: 郭朝斌

时长 13:39 大小 12.52M



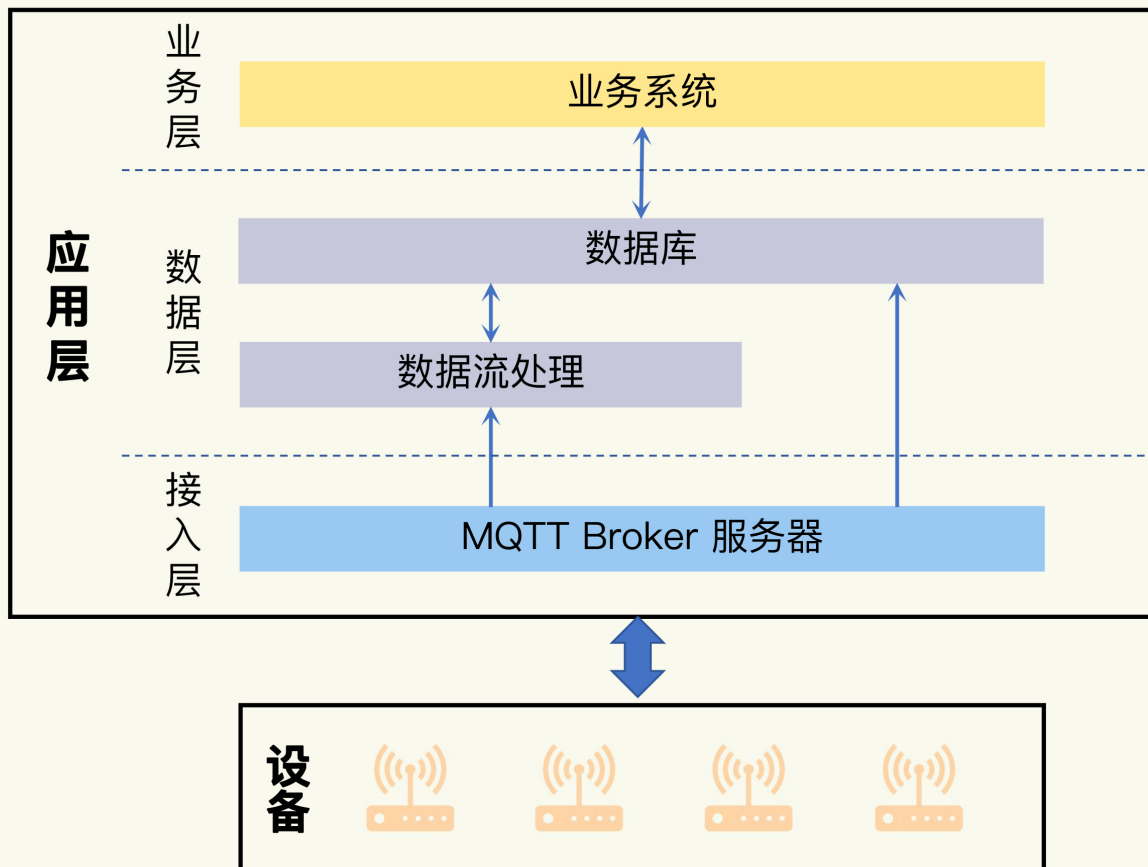
你好, 我是郭朝斌。

前面两讲, 我们一直在谈物联网云平台针对海量数据的处理和存储技术。顺着物联网的数据技术体系继续往下探索, 我们自然就会面临一个问题: 物联网云平台的服务器需要管理海量设备的接入, 并且接收来自设备的海量数据的输入, 那么服务器要怎么应对这样的挑战呢?

我们来分析一下这个问题。物联网设备是通过某种通信协议接入云平台的, 比如常用的 MQTT 协议, 那么设备接入的服务器就是 MQTT Broker 服务器。



从架构设计来说, 负责设备接入的这一部分一般叫作**设备接入层**, 也被称为 **IoT Hub**。设备接入层之后, 才是和互联网系统类似的**业务层**。具体的系统架构图可能是下面这样的:



不过，估计你也发现问题了，这样的架构根本无法支撑物联网场景中海量设备的接入和海量数据的输入。单台 MQTT Broker 服务器很容易面临性能瓶颈。所以前面提到的“服务器怎么应对挑战”的问题就变成了：怎么打造高性能的设备接入层？

我们再继续深入分析，把关键点定位得更具体一点。打造**高性能**的设备接入层，最重要的技术难点是，如何实现接入层的**高并发**。因为只有具备高并发的能力，才能有效地、可靠地实现数据的传输。

所以，现在问题又变成了：物联网云平台中，怎么实现高并发的设备接入层？

接下来，我就带你一步一步地设计一个高并发的设备接入层。在这个过程中，我会为你详细地讲解我们需要用到的重要技术，让你为物联网平台的搭建做好技术储备。

负载均衡：让多台服务器更好地协作

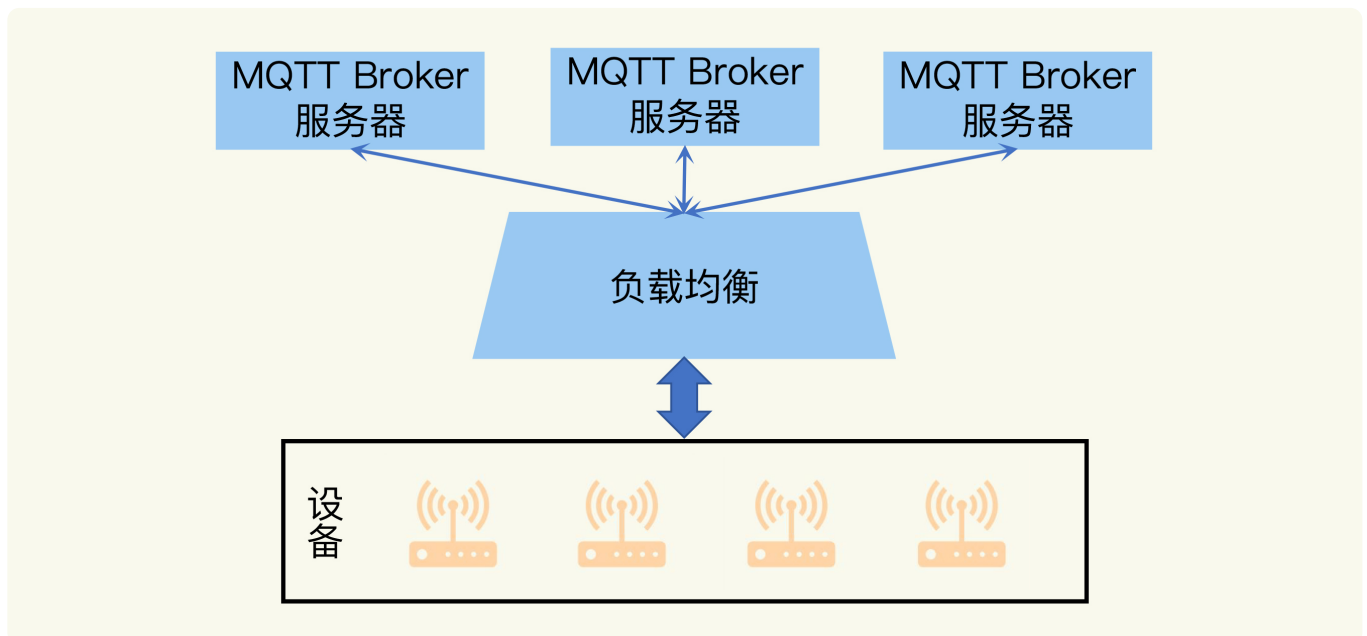
🔗 **上一讲**曾经我提到，为了解决灭霸这个难题，要让超级英雄们组成联盟。同样地，面对性能瓶颈，我们可以集合多个服务器的力量来解决问题，也就是说，可以通过增加多台 MQTT Broker 服务器来满足海量设备的连接请求。

但是服务器数量变多之后，它们具体要怎么协作呢？比如一个 MQTT 请求过来了，它应该被分配给哪台服务器处理呢？更进一步地说，怎么保证每台服务器的负担和压力都基本平衡呢？

这就要用到负载均衡技术。**负载均衡** (Load Balancer) 这个名字非常形象，我们可以分成“负载”和“均衡”两个词来理解。

负载是指服务器面对的网络连接和服务请求的**压力**，也就是“困难”。

均衡是针对服务器压力的解决办法，多个服务器一起来处理任务，并且这几个服务器的压力要达到**平衡**的状态，也就是“分担”。



这种平衡状态要怎么实现呢？我们需要设计一种任务**分配策略**。

最简单、直接的办法就是**平均分配**。

比如有 3 个服务器都在等待分配。当一个服务请求到来时，负载均衡服务器先将它给服务器 1 处理；接着第二个请求到来时，给服务器 2 处理，以此类推。

当第四个服务请求需要处理时，负载均衡转回来，再次指定给服务器 1，循环轮转。所以这种策略也叫**轮询策略** (Round Robin Scheduling)。

如果用公式来表达这个服务器序号的计算方法，那就是：

$$i = (i + 1) \bmod(n)$$

其中 i 是服务器的序号, 从 0 开始计数; n 表示服务器个数, 这里 n 等于 3。

当然, 在现实环境中, 各个服务器的配置并不是完全一样的, 有的任务处理能力强, 有的能力弱些。这就像在复仇者联盟中, 每位英雄的实力也各有不同。

我们自然希望能力强的服务器多负担一些任务, 所以对于轮询策略, 我们可以引入**权重系数**。

比如服务器 1 的权重是 2, 服务器 2 的是 4, 服务器 3 的是 1, 那么负载均衡服务器就可以给服务器 2 多分配任务, 而给服务器 3 分配最少的工作。

这个改良版的策略就是**加权轮询策略** (Weighted Round Robin Scheduling) 。

不管是轮询策略, 还是加权轮询策略, 它们具有简单实用的优点。但是通常情况下, 服务器处理每个任务的时间是不同。尤其是在物联网场景中, 设备可能会跟服务器保持长连接。可能有的服务器被分配的设备比较少, 但是保持连接的设备却很多。这种情况下, 如果我们仍然使用轮询策略, 那么各服务器的负载就很难达到平衡状态。

所以我们需要一种**动态调度**的策略, 能够基于各服务器的网络连接数情况, 优先将新的任务分配给保持连接的设备数最少的服务器。这种基于连接数的策略就是**最小连接数策略** (Least Connection Scheduling) 。

我们也可以为最小连接数策略引入权重系数, 给不同能力的服务器分配不同的权重, 更好地平衡服务器的负载, 这就是**加权最小连接数策略** (Weighted Least Connection Scheduling) 。

这个策略是怎么计算的呢? 假设服务器的权重分别是 W_1 、 W_2 和 W_3 , 服务器的连接数分别是 C_1 、 C_2 和 C_3 , 那么计算的表达式是这样的:

$$\text{Min}(C_1/W_1, C_2/W_2, C_3/W_3)$$

负载均衡服务器会选择计算结果是最小的那台服务器。

如果有 2 台或者 3 台服务器的计算结果同时最小呢？这时，我们还可以在这几台服务器中采用轮询策略。所以说，分配策略不是相互排斥的，在实践中我们需要综合使用不同的策略。

考虑到这个原因，我需要给你再讲一种常用的分配策略，那就是**源地址哈希值策略** (Source Hashing Scheduling) 。

所谓源地址，一般就是指网络连接的**源 IP 地址**。负载均衡服务器通过计算源 IP 地址的哈希值来确定对应的服务器。因为相同 IP 地址的哈希值是不变的，这就保证了相同的设备可以连接到固定的服务器上。

在实践中，我们可以通过开源软件来搭建负载均衡服务器。比如 **HAProxy 软件**，它支持 [🔗 OSI 网络七层模型](#) 中第 4 层和第 7 层的负载均衡，性能可以跟商用解决方案媲美。同时，我建议搭配 **Keepalived** 软件使用，实现高可用的热备方案，这样就可以避免单机故障导致系统瘫痪了。

消息队列：避免耗时的等待

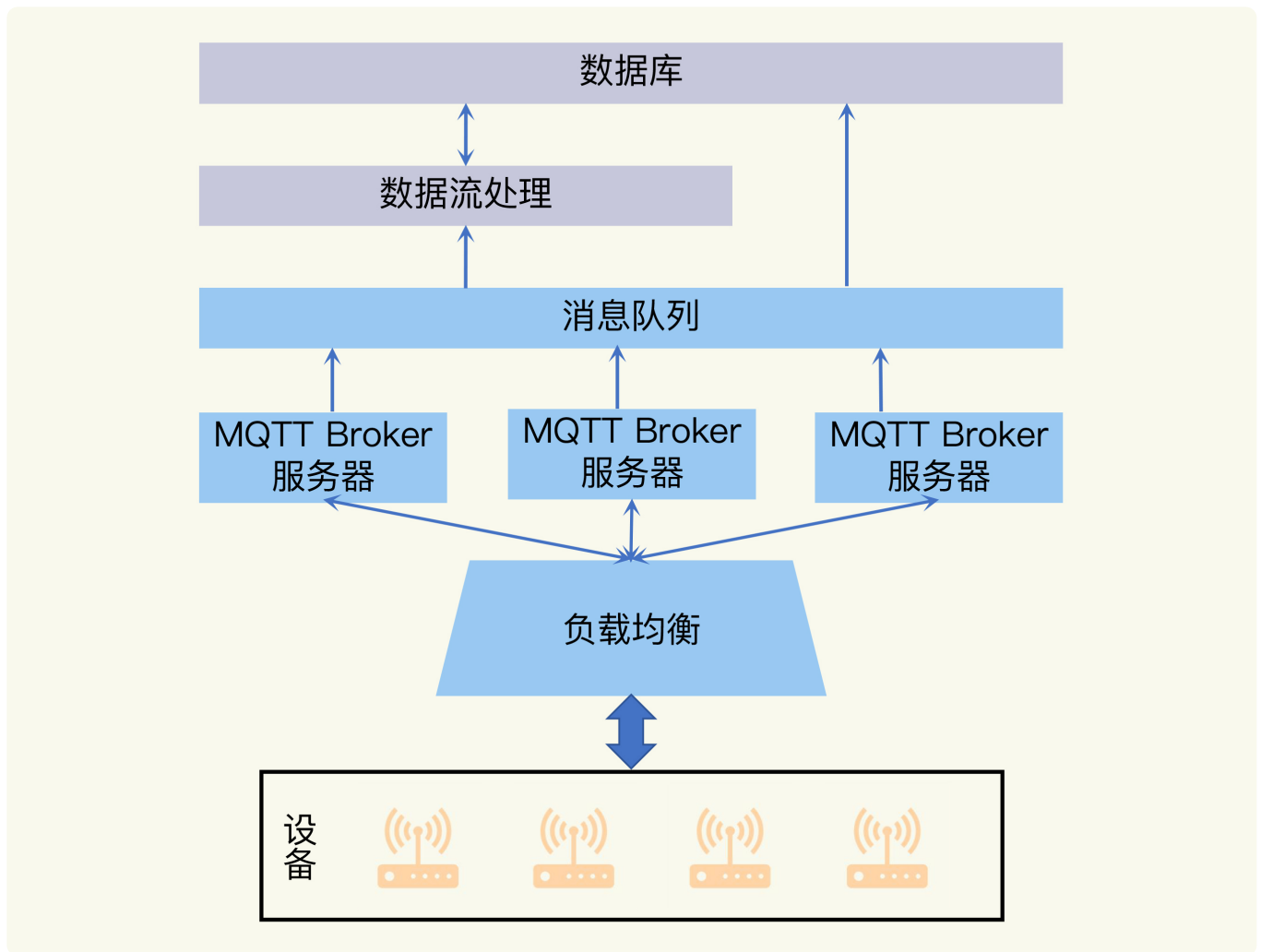
当负载均衡服务器将物联网设备的网络请求分配到 MQTT Broker 服务器后，MQTT Broker 服务器就可以与设备建立连接，并且收到设备上传的数据了。然后，数据就会传输给数据流处理服务器或者写入数据库中。

但是，数据流处理和把数据写入数据库都是花时间的操作，它们和数据传入的速度并不一致。为了可以让 MQTT Broker 服务器高效地完成数据的传输，同时保证数据流处理和写入数据库的操作可靠执行，我们需要在 MQTT Broker 服务器和数据流处理服务器之间加入**异步处理**机制。

那么，异步处理机制要怎么实现呢？行业内一般通过**消息队列**来实现。你可以把消息队列想象成双十一购物节后的快递公司，它收到商家大量的运单，然后快递公司按照一定的节奏完成这些运单的递送任务。

在我们的系统中，MQTT Broker 服务器将数据给到消息队列，就完成了数据传输的工作。在这之后，数据流处理和写入数据库的操作只要按照自己的节奏，消费消息队列内的数据

就行了。



另外，消息队列还提供了额外的两个好处：

1. 实现了 MQTT Broker 服务器和数据流处理服务器之间的解耦，双方没有直接的依赖，所以维护更新会更加方便。
2. 可以平衡输入数据量的大小变化，所以数据流处理服务器不会因为骤增的压力而崩溃。

既然消息队列这么重要，你肯定也想知道，在实践中有没有开源的软件可以选择呢？答案是有的。最常见的选择有 **Kafka** 和 **RabbitMQ** 等。

Kafka 真是一个神奇的软件，凭借着优秀的设计成为消息队列系统的主流选择。而且在这个成功的基础上，它还在不断改进，新增了分布式流处理和分布式存储等功能。虽然这些新功能不一定比专门的流处理和存储软件更强大，但对于中小型的业务来说完全够用了，关键是非常方便，部署也简单。

缓存系统：让数据读写更快速

从架构图中我们可以看出来，数据流处理服务器处理完数据后，这些数据会存储到数据库中，提供给批处理或者业务服务器使用；而且它在进行数据处理时，也需要从数据存储中获取一些信息。

但是数据库的读写数据是一个速度比较慢的操作，尤其是基于磁盘介质存储的数据库。为了提高性能，我们需要比数据库，甚至分布式文件系统更快的数据存取方式，这就需要用到**缓存系统**了。

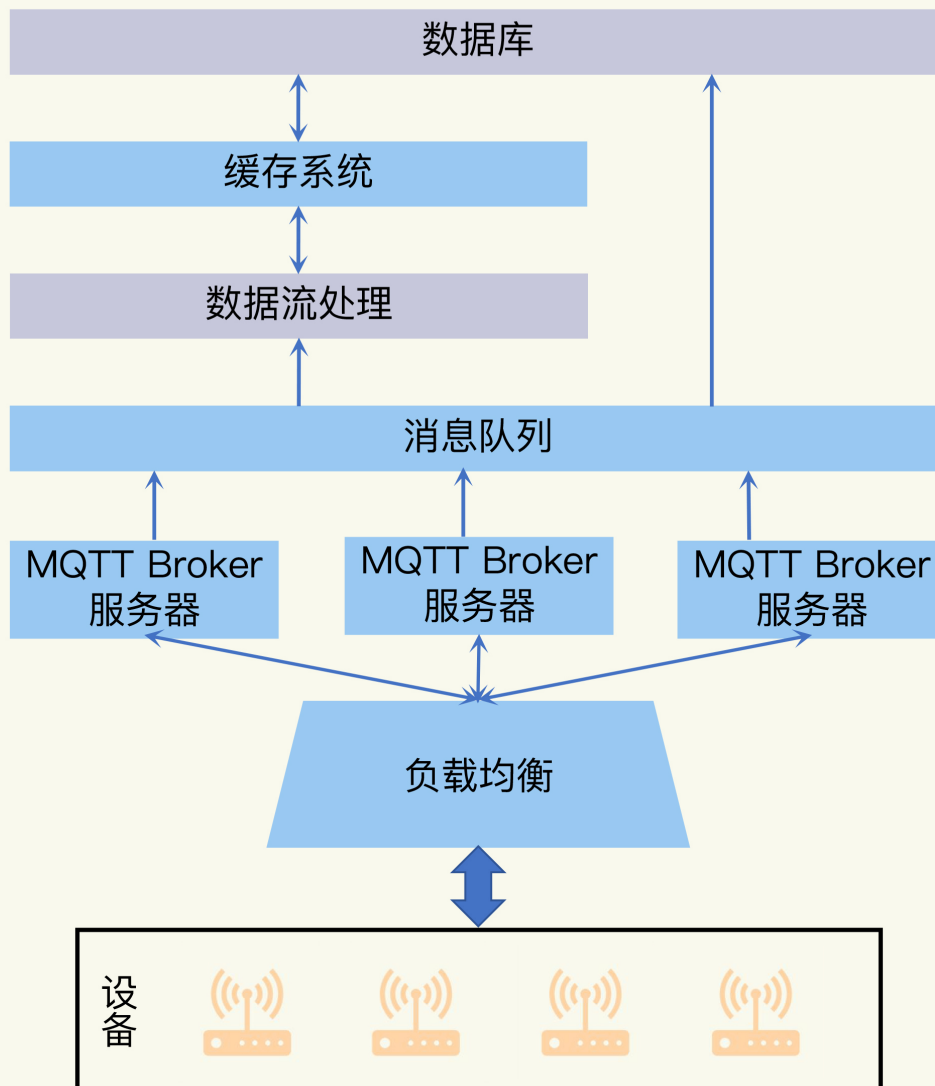
缓存是一个常见的概念。比如浏览器会使用缓存来避免重复从网络获取网页数据，从而更加快速地响应用户的请求。

在我们的系统中，引入缓存系统当然是为了避免直接从磁盘中读取数据，或者直接向磁盘中写入数据。

缓存系统一般会将数据暂时存储在内存中，这样数据流处理应用就不需要直接与低速的磁盘打交道了。而且，如果我们读取的是经常用到的热点数据时，这些数据全都不需要重复从磁盘读取。这样既减轻了数据库的压力，又提高了数据处理速度，一举两得。

缓存系统的常用开源选择有 **Redis** 和 **Memcached** 等。其中 Redis 更是在数据持久机制和主从节点复制的高可用特性上做了很多工作，不但功能强大，而且效率也很高。

加入缓存系统之后，设备接入层的整体系统架构就完成了，你可以参考下面这张图。



小结

总结一下，物联网云平台的服务器为了应对海量设备接入和海量数据输入的挑战，需要打造高并发的设备接入层。所以在这一讲中，我带你设计并完善了设备接入层的整体架构，并讲解了需要用到的**负载均衡**、**消息队列**和**缓存系统**等技术。

1. 负载均衡用来协调多台服务器来共同应对网络连接和请求的压力，服务器任务分配策略主要有三大类，分别是平等分配的轮询策略，考虑连接设备数的最小连接策略，以及保证相同的源地址访问同一台服务器的源地址哈希值策略。
2. 这些策略可以搭配使用，还可以通过引入加权系数来改进。在实际应用中，我们需要根据场景灵活选择，常用的开源方案有 HAProxy 软件和 Nginx 软件，部署时可以搭配 Keepalived 提高可用性。
3. 消息队列可以在不同的系统之间搭建桥梁，保证数据和服务请求的高效可靠处理。常用的开源软件有 Kafka 和 RabbitMQ 等。

4. 缓存系统可以减轻数据库的压力, 提高系统响应速度。常用的开源软件有 Redis 和 Memcached 等。

思考题

最后, 我想请你思考一个问题。

在这一讲中, 我们通过负载均衡来分担服务器的压力。这里设备是使用 MQTT 协议与接入层服务器, 也就是 MQTT Broker 服务器通信的, 所以不同的设备发送的相同 Topic 的消息就会发送到不同的服务器上。

如果有订阅者订阅这个 Topic 消息, 那么应该怎么保证订阅者可以接收到所有的设备发送的此 Topic 的消息呢?

欢迎你在留言区写一下自己的思考, 同时也欢迎你将本讲分享给对高并发感兴趣的朋友一起讨论学习。

提建议

© 版权归极客邦科技所有, 未经许可不得传播售卖。页面已增加防盗追踪, 如有侵权极客邦将依法追究其法律责任。

上一篇 11 | 数据存储: 物联网中的数据库有哪些?

精选留言 (7)

写留言



Geek_9b7997

2020-12-04

一般的企业级平台毕竟不是大厂平台, 如果有了边缘网关后, 并发量是不是就没有那么多了? 其实也无需集群了?

展开



**稳**

2020-12-05

请教下老师，mqtt不也属于消费队列吗？那增加的rabbitmq这一层的意义大吗？

**fwx**

2020-12-04

当接入 IoT 设备数量在十万以内或者万级时，能否直接选择支持MQTT协议的消息队列，将MQTT Broker服务器和消息队列系统进行合二为一？

展开 ∨

**il李li**

2020-12-04

对于每个设备负载均衡分配连接到那个MQTT Broker，收到设备PUB消息的Broker路由转发该消息到集群的其他Broker，这需要集群中维护一个全局的主题Topic与消息路由表Route Table的对应关系，能够准确转发到其他有订阅该主题的Broker节点上。而对于Broker收到PUB消息后，查到相应的Topic，并将消息发送给订阅此 Topic 的终端设备。这个全局的Topic消息路由表是个关键的设计

展开 ∨

**贤伟**

2020-12-04

我们是用3 kafka brokers集群做消息处理的，每个topic有多个分区，每个分区有3个副本，分在3个broker上。同时每个topic我们有一个消费者group，每个group 有多个消费者，每个消费者去处理1个或多个partition（消费者就是数据处理微服务，可以动态扩展，保证启动时订阅同一个topic，属于同一个group）。

我的理解可以把consumer group当做一个订阅者，既能利用多分区并行消费，也可以...

展开 ∨

**德慢慢**

2020-12-04

可以利用tag吗？

展开 ∨





wugily
2020-12-04

利用kafka 的group和topic?

展开 ▾

